XML IN WEB-BASED TRAINING: WORKFLOW AND OPPORTUNITIES

Simon Wiest and Andreas Zell

ABSTRACT

In this paper we point out the benefits of using structured content in Web-Based Training (WBT) scenarios. Using an XML compliant subset of the DocBook document type definition – enriched with semantics for interactive hypermedia and educational elements – we present a complete publishing workflow for educational online material. Original content is authored in an SGML/XML editor, then processed into a set of XML files containing course structure and contents. Finally a web server extension written in Java (servlet) renders this XML content on-the-fly into HTML pages according to learner preferences, desired target media or current educational situation. An XML content base also makes way for a fine-grained annotation facility. Finally, all user requests and their parameters are logged into a relational database. This allows the collection and sharing of comparable data on applicability and effectiveness of the content and helps to improve the quality of future online courses.

KEYWORDS

Web-Based Training; WBT; educational media; XML

1 Introduction

The increasing availability of networked personal computers and the rise of the World Wide Web (WWW) in the mid-90s gave birth to Web-Based Training (WBT). Online delivered courses over the WWW seem to address very well some of the problems faced in traditional CD-ROM production. Hypertext Markup Language (HTML), as the *lingua franca* of online content, has become a common storage data format and offers built-in semantics for linking several content modules due to its hypertext capabilities. The task of shipping new versions can now be reduced to the maintenance of one single version stored on a centralized web server. The network connection between the learner's client computer and the course web sever offers new possibilities for tracking content usage and user progress.

Unfortunately, as it turns out, HTML lacks much of the functionality needed to fulfil many of the needs of online educators, e.g. no dedicated semantics for online exercises, no support for complex equation rendering, difficult reformatting of HTML content that is to be embedded in new contexts.

This paper will present a workflow that tries to tackle some of these problems using a non-proprietary content structure. It also introduces a web server extension that enables HTML browsers to access the features made possible by XML content. The paper is structured as follows: Part 2 suggests how structured content can influence the online learning process, Part 3 gives an overview of selected SGML/XML applications in this field. Part 4 describes our own implementation of an XML-based publishing workflow tailored to the needs of online training. Part 5 then introduces new features made possible by an XML content base. Finally, Part 6 will discuss the implications of our work for usage evaluation and content improvement.

2 Benefits of structuring educational content

The key idea behind structuring educational material is to decouple style from content. Storing course contents in elements that reflect the semantics needed in learning applications (e.g. introduction, example, proof, quiz, ...) allows the reuse of data in different situations and formats - adapted to learner preferences, target media or purposes. All groups concerned with the online learning process benefit from the use of structured documents:

Authors

- Authors can concentrate on content not (necessarily) on style and formatting.
- Structural integrity can be checked/enforced by an authoring application (e.g. hyperlinks can be validated automatically)
- Hyperlinks can point to document fragments, allowing fine-grained cross-references.
- Indexes, summaries, glossaries etc. can be generated automatically.
- Content modules can be reused in other contexts (e.g. related courses, in a guided tour, in different skill levels).

Publishers / Internet Service Providers

- Content is machine-processable (i.e. easily stored in databases to facilitate content administration and improvement of retrieval performance).
- Multi-target publishing is possible from a single source (e.g. online/offline versions, printed material). The concept of *viewgroups* presented in [13] addresses this idea.
- Structured content allows the attachment of meta-data not only to the complete document but also to its subcomponents.
- Versioning is facilitated.

Learners

- Mandatory structure facilitates navigating the course content.
- Advanced search and retrieval of content components.
- New features like multi-representation content become feasible (e.g. the same content in both visual and textual representation. Ram *et al* suggest a Presentation Markup Language for this purpose in [15].)

Researchers / Quality Assurance

- Enhanced possibilities for learner tracking and usage evaluation allow the collection and sharing of comparable data on applicability and effectiveness of the content and help to improve its quality [5].
- Flexible rendering allows experiments with various screen designs, navigation concepts etc.

3 Benefits of using XML

Like its predecessor, SGML [11], XML [6] is a scheme to define new languages. It does not specify what kind of application data is stored in a file by itself but rather provides a framework to define purpose-depended languages that follow one common syntax.

This is done in a document type definition (DTD) that describes the overall structure and the components of generic data file. In an ideal world two applications dealing with the same class of content (e.g. images, text documents, invoices, online-courses etc.) would share the same DTD. This is, unfortunately, not the case in the real world. Nevertheless, if both applications store their data in an XML compliant structure, developing a converter between these two formats is much easier and less error prone than a translation between two binary formats.

While the interpretation of the actual data within XML documents depends on the type of application, XML structures can be processed by the same parsers and tools with both commercial and free implementations [1], [2] available.

There have been already different suggestions to apply SGML, XML or at least structured documents to educational purposes.

The IEEE Learning Technology Standards Committee proposes a learning object meta-data standard (LOM) [10] to facilitate the retrieval, reuse and exchange of educational content. Learning objects range from small components like a single sentence, a figure or a Java applet up to complete online courses. LOM contains information on technical, didactical and legal issues of educational content and provides support for versioning. Beside the MIME type of a learning object it does not include any information about how the object should be rendered, executed or interpreted.

The Tutorial Markup Language [7] provides a language to describe several classes of online exercises like multiple-choice tests, drag & drop puzzles etc. Other structured file formats [4], [12] persue the same goal but currently use neither SGML nor XML.

The Synchronized Multimedia Integration Language (SMIL) [9] allows an XML compliant specification of both time-line based and event driven multimedia presentations. Supported by widely distributed player applications (e.g. Real Player from Real Networks, Apple Quicktime 4.1, limited support in Microsoft Internet Explorer 5.5), it definitely will be used in educational context in the future although it does not contain dedicated didactic semantics.

4 Our Implementation

At our department, we offer several online courses on computer science topics to our students. This material consists of some 200 linked HTML pages per course, enriched with additional interactive elements like Shockwave animations, Flash movies, and Java applets. The courses were created from legacy content in Adobe FrameMaker format as a starting point. Dealing with mathematical and statistical subjects the courses contain over 1000 equations. A typical page is shown in Fig. 4.

Although FrameMaker ships with HTML/XML export functionality, crucial parameters (like the file names of resulting HTML and image files) cannot be controlled as precisely as necessary. Even worse, the files generated by the HTML export function needed manual clean-up which took one person about 1-2 days to complete for each new release of a course. Additionally, structural navigational aids have to be imposed manually to make the resulting HTML files usable for learners.

We therefore came up with a new publishing workflow that consists of six steps:

1. Choosing/creating an appropriate document type definition (DTD).
2. Structuring a complete course according to the DTD in an authoring environment.
3. Saving the structured content in XML format and preparing it to be served on the WWW.
4. Creation of HTML page templates and navigational controls.
5. Developing a servlet that converts XML to HTML on-the-fly.
6. Tracking content usage into a relational database.

4.1 Document Type Definition (DTD)

In the search for an appropriate document type definition we realized that while there are quite a few standard DTDs for technical documentation like DocBook [14], our course content structure had also to support didactic and hypermedia elements. We decided to enrich an XML-compliant, light-weight variant of the DocBook DTD with additional elements and attributes that cater for educational needs, different media types and hyperlink capabilities.

Educational elements: According to the idea of learning objects proposed in [10], educational content can be decomposed into subcomponents of different aggregation level:

Table 1 LOM aggregation levels

Level	Example
0	"content atoms" like single sentences, images or raw media files
1	A collection of level 0 objects like a single HTML page with some embedded pictures
2	A collection of level 1 objects like a linked net of HTML pages with common index page
3	A collection of level 2 objects like a complete course

We classified the elements inherited from DocBook into the given aggregation levels appropriately. All level 0 objects were assigned a 32digit hexadecimal identifier attribute `id`, unique in time and space. All level 1 objects also got an attribute `time`, which reflects the assumed learner's time to complete this object. A typical level 1 object is a short quiz which consists of a question and an answer.

Further educational elements are `experiment`, `procedure`, `definition`, `algorithm`, `proof`, `example`, `important`, `note`, `aim`, `didactical`.

Media type support: We added several elements for different media types with appropriate attributes specific to the media type. Media elements are `graphic`, `equation`, `applet`, `shockwave`, `flash`, `quicktime`.

Hypertext support: Hyperlinks are defined according to a subset of the W3C XLink working draft of July, 26 1999 [8].

4.2 Authoring environment

We employed FrameMaker+SGML for XML editing because it combines the ease of use of a modern word processor (especially equation handling) with the structural functionality of an XML editor. It also facilitated the use of legacy content stored in FrameMaker files. To track down learning objects during the following processing steps, each level 0 object was assigned a globally unique identifier. We implemented this feature by a FrameMaker+SGML API plug-in that generates an identifier by mangling date, time and the MAC address of the computer's network adapter into a 32 digit hexadecimal number unique in time and space. On Windows systems, the RPC UUID library functions are used for this. Once the identifiers are assigned,

they will be present in FrameMaker+SGML files as well as in exported SGML documents. It is important to note that even though contents are kept in the FrameMaker+SGML file format, this format represents nothing more than a wrapper around a DTD compliant structure.

4.3 Converting and preparing XML content

The next step is to export the FrameMaker contents into an SGML file using built-in export functionality. Image representations of embedded equations are also generated in this step.

A common problem creating scientific content for the WWW is to display mathematical equations. While some browsers already feature limited MathML rendering capabilities it is common practice to embed equations as bitmap images, e.g. as GIF or PNG files. This one-way conversion complicates later-on modifications for the author and results in poor image quality of the content (especially when hardcopied).

FrameMaker is capable of exporting and importing structured textual representations of equations. We decided to keep equations in this format within FrameMaker but to convert them to GIFs during SGML export. By exporting the course contents twice with different resolution (72 dpi, 300 dpi), we get two versions of the same equation. Using the unique IDs the corresponding image files can be determined. Their filenames are stored in the SGML content file as attributes of the `equation` element. This allows features like zooming into equations that are hard to read (see Fig. 4 right) or high quality hardcopy options.

In a final step the *course description*, a hierarchical table-of-contents structure is extracted from the SGML document to serve as a Web-Site structure in the following steps. The SGML file is then broken down into a number of XML files, each containing a subchapter of the course with a typical length of 1-2 screens. This step is done to improve the web server response time and might be replaced by a database approach in the near future.

Course description, course content (now stored in a set of XML files) and equation images are now copied onto the web server file system, ready to be served. The complete conversion procedure is depicted in Fig. 1. Export and equation rendering of a 200 page course on a PII 350 MHz Windows PC takes roughly 10 minutes for the 72 dpi version (40 minutes for 300 dpi) and does not require any human interaction.

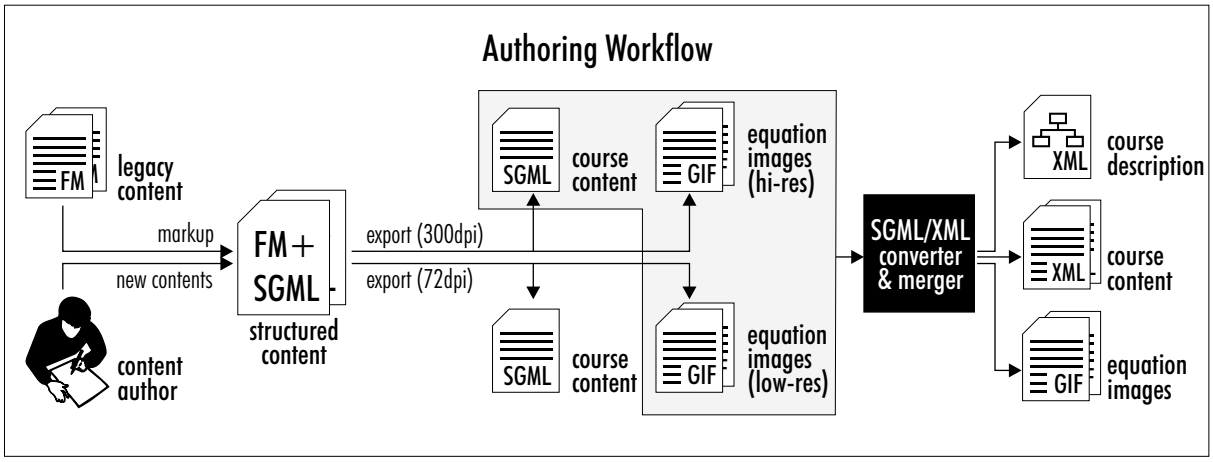

Fig. 1. Authoring workflow. Content creation takes place inside the SGML editor (like FrameMaker+SGML). In two passes, SGML versions and two variants of equation graphics are exported. Finally these files get converted to XML and prepared for being served by our servlet-based system.

4.4 Page templates and navigational controls

Page templates were designed in a HTML editor, containing placeholders for dynamic content (like a site tree on the left, navigation controls and a content path at the top and a main content area at the right). These placeholders get replaced by applying several XSLT transformations [18] using an open source XSLT formatter [2] to the requested course content file. Because of the complex XSLT semantics, transforming the source XML content consumes the most processing time (>70%) of serving a HTTP request.

XSL style sheets exist for several target media (online, print) and usage modes (tutorial and script mode). Fig. 3 shows three examples of different modes and target media rendering.

The globally unique learning object identifiers are preserved during conversion, allowing web browsers to make use of this information, e.g. for a fine grained annotation facility (see below).

4.5 Serving content

Because most browsers currently provide only limited XML support, the course contents are server-side converted from XML to HTML. Additionally, this allows adaptation of the contents to a specific user's learning context (e.g. formatting preferences, target media, usage mode) at this point.

To bridge the gap between web server and XML content, we developed a *S*ervlet *I*nterface for *O*nline *U*ser-adapted *X*ML (SIOUX). Servlets [16] are plug-in extensions written in Java that add functionality to a HTTP daemon. In our case, URL requests homed on XML files get redirected by the HTTP daemon to SIOUX which

a) reads in the requested XML file,
b) applies a XSL transformation to the target medium and mode,
c) inserts the content into a page template,
d) resolves server-side includes and other placeholders,
e) returns the HTML page to the learner's client and
f) logs user id, URL and learning context into a relational database.

Our setup builds on an Apache HTTP daemon [3] and the MySQL RDBMS [17] running on an RS/6000 workstation. Servlet support is provided by the JServ servlet engine, connecting to the SIOUX servlet on a Windows PC (Windows NT4.0, Pentium II-350, 256 MB, Sun Java Runtime Environment 1.2.2). Running HTTP daemon and the SIOUX servlet on two separate workstations seems to slow down response times due to the necessary network overhead, but tests revealed that the advantage of load balancing on two processors compensates this by far.

The response times vary between 0.2-1.0 seconds per page request depending on complexity of the XSLT style sheets. This is acceptable compared to the 1-5 minutes it takes a learner to work through the page contents.

Fig. 2 shows how a request is handled by Apache/SIOUX and which resources are used.

4.6 Tracking content usage

The system contains a learner database that allows restricting access to course contents, personalized tracking of content usage and persistent storage of user preferences. After a request has been served, the user's ID, the requested URL and all session parameters (e.g. current mode, target media, individual interface settings, date and time of access, referring URL, agent, ...) are logged into a relational database.

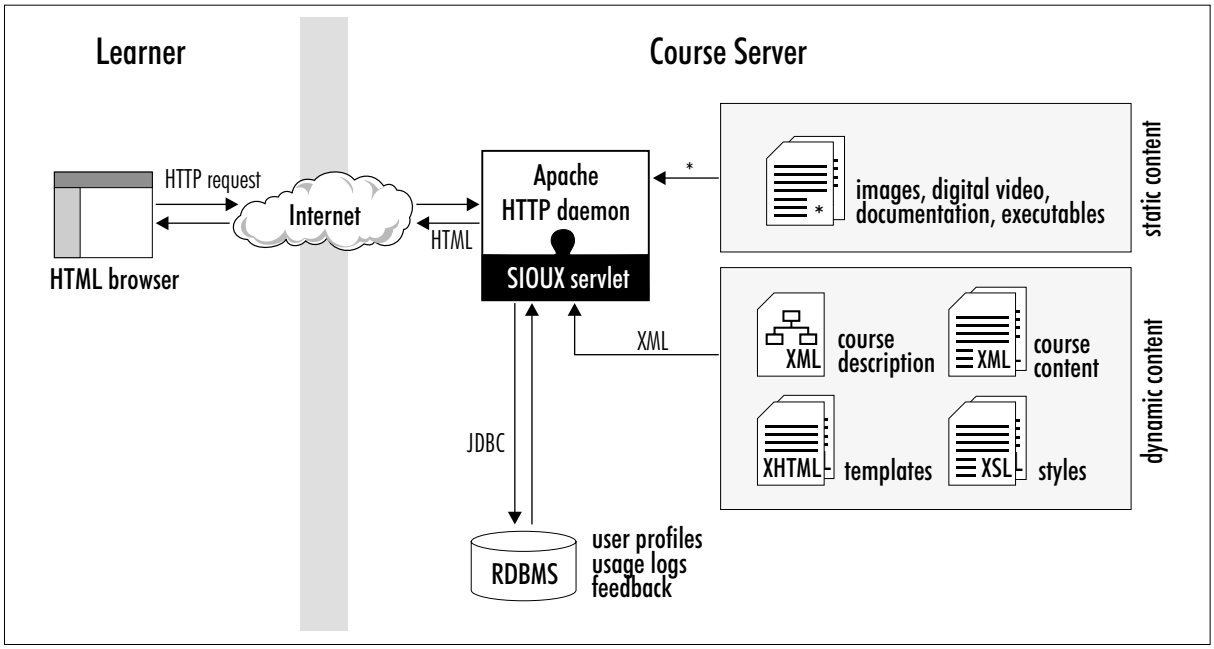

Fig. 2. Resources used within the SIOUX architecture. The SIOUX servlet generates dynamic content from XML structure, content, style and template files. Static content is handled by the Apache server without SIOUX intervention.

5 New features made available by the XML content base

Migrating to an XML content base allowed us to implement three new important features.

5.1 Different learning modes from one single content

From our previous experience we could identify two main usage scenarios for our online content: the *tutorial mode*, in which learners tend to navigate through the course linearly and slowly paced and the *script mode*, in which users want to look up small amounts of information like definitions, proofs or concepts quickly.

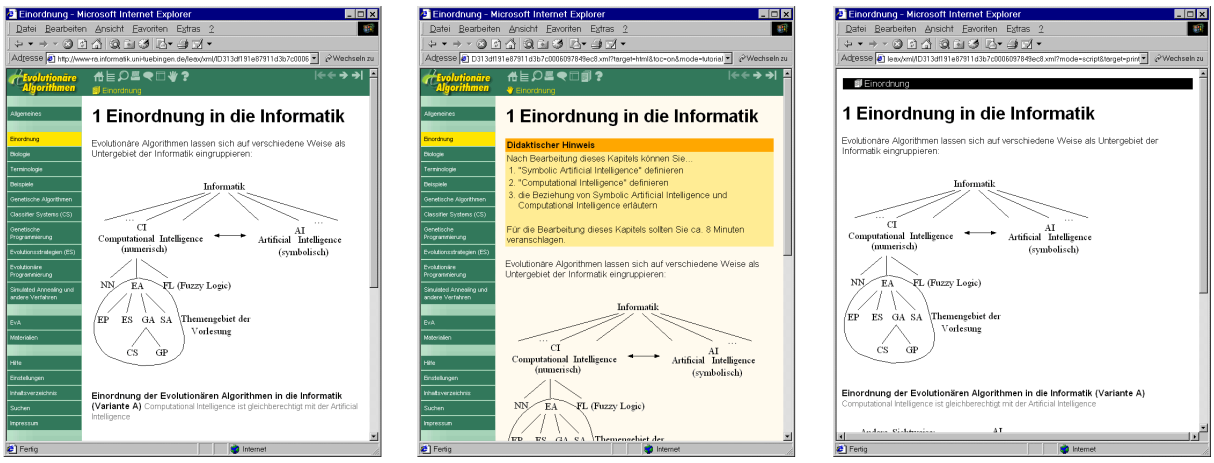

Fig. 3. The same XML content rendered for different modes/targets. Left: Script mode for on-screen usage. Middle: Tutorial mode for on-screen usage. Note the box containing learning goals in the middle of the main content area. Right: Script mode for printout. It has higher resolution and contrasts that enhance printability. Navigational elements are suppressed because they don't make sense on printed paper.

While didactic instructions (“It will take about 30 minutes to complete this chapter.”) make sense in the former mode, they are useless in the latter and should be suppressed.

Because both modes are just different representations of the same content, we wanted to keep maintaining course contents simple and be able to generate both versions from one single data source. Because interface design for WBT is still a young discipline we also realized that there might be a need for additional modes in the future. Ideally, a learning environment therefore applies formatting and user preferences to the desired content not until the moment of an incoming learner’s request. An additional print view offers also a high quality hardcopy option.

Fig. 3 depicts variations in representation according to different modes and targets.

5.2 Improvement of equation handling

An evaluation among student users showed that the poor display quality of equations in HTML documents was a major annoyance to learners. Usually rendered at a resolution of 72 dpi, small indices and other mathematical symbols become hard to read on screen. Using XML it is easy to relate one document element (like an equation) with multiple representations of its content, e.g. graphic files in low (72 dpi) and high (300 dpi) resolution or MathML trees. Using our improved system, learners can click on any equation to open a high resolution version in a separate window (see Fig. 4, right).

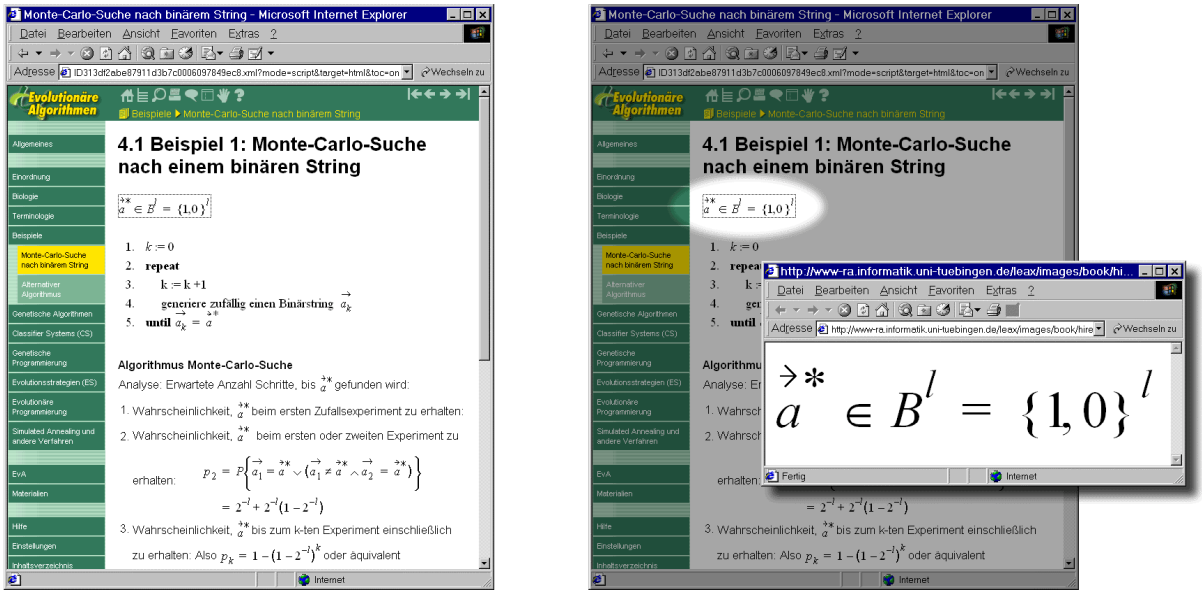

Fig. 4. Left: Typical page from online course “Evolutionary Algorithms”. Note the high number of equations and symbols in the main content area. Small symbols, especially indices, are hard to read on-screen and do not print very well. Right: Users can enlarge any equation by simply clicking on it.

5.3 Fine-grained annotation facility

To support cooperative work many learning environments offer electronic bulletin boards (e.g. newsgroups) related to the course content. While this allows users to share their questions and comments about the learning subject, discussions take place separated from the content itself. Even attaching comments directly at the end of an HTML document might require clumsy references to the element being annotated (“What does variable x stand for in the 3rd equation of the 7th paragraph?”).

Retaining the unique element IDs from the XML during conversion into HTML makes way for a smart, fine-grained annotation system that works at the subcomponent level within HTML pages. Using Internet Explorer, the learner can comfortably mark a certain spot in the online document and add a comment right there. Users of other browsers (like Netscape) are shown “comment hooks” (small gray squares) grouped at the end of each element that can be annotated. A typical annotation sequence is shown in Fig. 5.

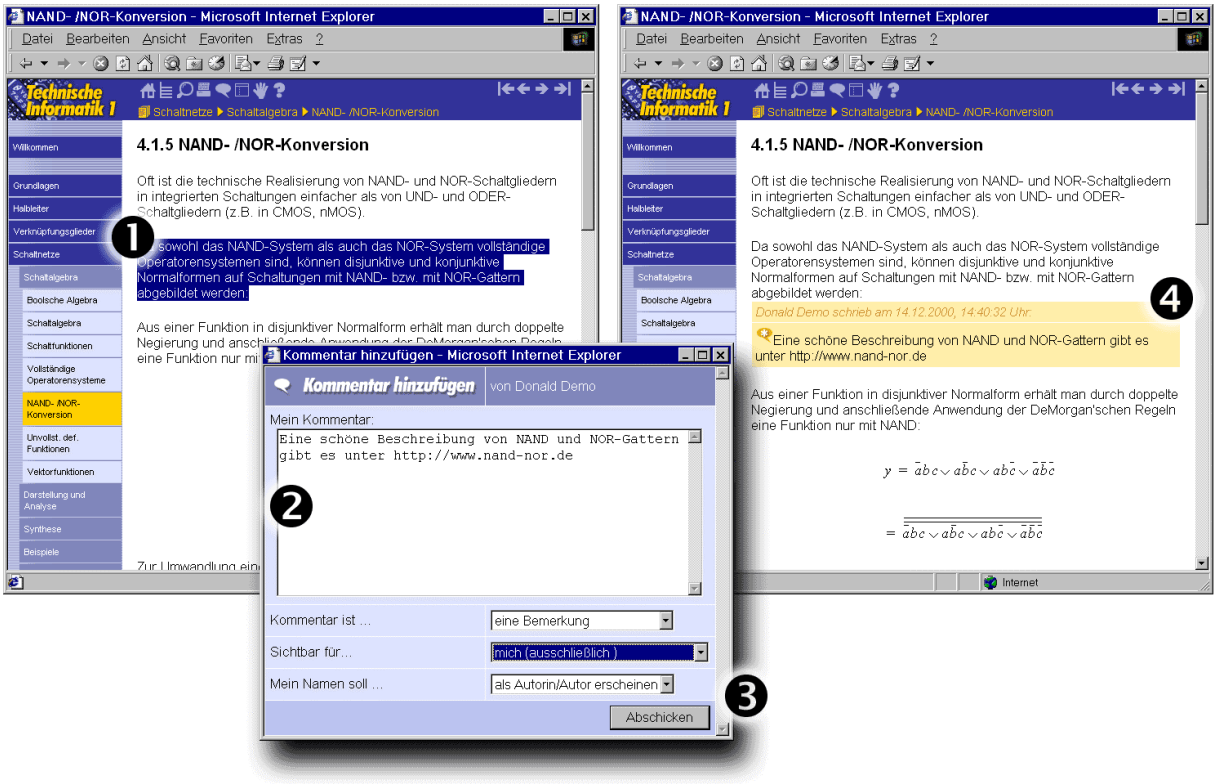

Fig. 5. Annotation sequence: 1.) Selection of some text or graphic to be annotated. 2) Annotation is entered in a pop-up window. 3) Classification and submission of annotation. 4) Reloaded document displays annotation embedded within the document.

Users can classify their annotation in one of four categories (general comment, bug report, question, answer) and limit its visibility for other users (no restrictions, visible only to learner and course tutors, visible only to learner). Finally, to encourage the sharing of comments learners may choose to remain anonymous when posting annotations, e.g. if they are afraid of submitting a “silly” question.

Because annotations are linked with a specific element and not with the HTML document, URL or course chapter, they stay in correct position after editing the original course contents, even during re-use of a document in the context of another course.

6 Conclusion

We have illustrated how structuring content supports the online learning process in creating, deploying and using course content. Examples have been given of how existing DTDs can be modified to address the specific needs of online educators. Using this XML compliant DTD offers dedicated semantics needed in didactical processes while avoiding getting trapped in a

proprietary format. The publishing time of new course releases could be cut down from 1-2 days of manual labor to some minutes batch processing time.

Using one single source which is varied on-thy-fly avoids complicated and error prone maintenance of several parallel versions. Rendering is sufficiently fast for online learning purposes. Several modes (tutorial, script) match the main usage scenarios closer than one, single all-purpose version. The handling of equations has been improved by providing high resolution versions for screen and printout. The equation itself remains in an editable textual representation which might use MathML in the near future.

Preserving learning object identifiers in the resulting HTML pages makes way for applications like fine-grained annotation mechanisms on sub-page detail level if browsers provide scripting access to the HTML source code (like Microsoft's Internet Explorer using JavaScript).

The flexibility inherent in displaying content in different styles provides a test bed for the evaluation of different user interfaces or navigation controls, because only the page templates, not the content itself, have to be changed. Equally important is the ability to measure the usage of these experimental variations. While it is obvious that web interface design plays an important role in online education processes, it is arguable which is the *best* way. Due to extended tracking possibilities we can offer an approach that helps to collect hard facts. In the likely case we will learn that there is no single ideal interface for *all* learners, the presented workflow is already powerful enough to serve highly personalized educational content. In any way, precise usage data will help WBT authors to understand how students are using their course content as well as web interface designers to improve their work based on learners acceptance of interface innovations.

Acknowledgements

The site description file format has been originally developed by Igor Fischer. The research described in this papers was conducted within the project "Bioinform@tik", funded by the state of Baden-Württemberg and the Deutsche Telekom AG.

References

1. Apache XML Project, "Xerces Java Parser," http://xml.apache.org/xerces-j/index.html
2. Apache XML Project, "Xalan-J XSLT," http://xml.apache.org/xalan/index.html
3. Apache Software Foundation, "Apache HTTP Server Project," http://www.apache.org/httpd.html
4. Arneil S., Holmes M., and Street H., "Hot Potatoes," University of Victoria Language Centre, http://web.uvic.ca/hrd/halfbaked/
5. Barquero B., Creß U., and Hesse F.W., "BioInform@tik. Evaluation der multimedialen Lehrveranstaltungen im Sommersemester 99," internal report, Deutsches Institut für Fernstudienforschung an der Universität Tübingen, University of Tübingen, Tübingen, 1999.
6. Bray T., Paoli J., and Sperberg-McQuee C.M., eds., "Extensible Markup Language (XML) 1.0," W3C Recommendation 10-Feb-1998, http://www.w3.org/TR/1998/REC-xml-19980210.html
7. Brickley D., "Tutorial Markup Language (TML)," Institute for Learning and Research Technology, University of Bristol, http://www.ilrt.bris.ac.uk/netquest/about/lang/
8. DeRose S., Orchard D., and Trafford B., eds., "XML Linking Language (XLink)," World Wide Web Consortium Working Draft 26-Jul-1999, http://www.w3.org/1999/07/WD-xlink-19990726
9. Hoschka P., ed., "Synchronized Multimedia Integration Language (SMIL) 1.0," W3C Recommendation 15-June-1998, http://www.w3.org/TR/REC-smil/

10. “Draft Standard for Learning Object Metadata,” IEEE Standards Department, Learning Technology Standards Committee, Piscataway, 2000. (IEEE P1484.12/D4.0).
11. “ISO 8879: Information processing – Text and office systems – Standard Generalized Markup Language (SGML),“ International Organization for Standardization (ISO), Genf, 1986.
12. Kraußе R., “Exercise Format,” TU Dresden, http://linus.psych.tu-dresden.de/Stupla/ef/
13. Müller H., Deponte J., “Distributed Multimedial Presentation and Conferencing,“ Proc. Workshop “Die Virtuelle Wissensfabrik“, GMD, St. Augustin, 1999.
14. Organization for the Advancement of Structured Information Standards (OASIS), “The DocBook DTD,“ http://www.oasis-open/docbook, 1998.
15. Ram A. *et al.*, “PML: Adding Flexibility to Multimedia Presentations,” IEEE Multimedia, Vol. 6, No. 2, April 1999, 40-51.
16. Sun Microsystems Inc., “The Java Servlet Specification v2.2,”, December 1999, http://java.sun.com/products/servlet/
17. T.c.X, DataConsultAB, “MySQL,” http://www.mysql.com
18. World Wide Web Consortium (W3C), “Extensible Stylesheet Language (XSL) Version 1.0,” W3C Working Draft 27-Mar-2000, http://www.w3.org/TR/xsl

Simon Wiest and Andreas Zell
Universität Tübingen
Wilhelm-Schickard-Institut für Informatik
Sand 1, 72076 Tübingen
Germany

Email: {wiest, zell}@informatik.uni-tuebingen.de